Chapter-6

A NOVEL SYNERGIC APPROACH OF NEAREST NEIGHBOR TECHNIQUES WITH GENETIC ALGORITHM FOR CLASSIFICATION
Chapter 6: A Novel Synergic Approach

This chapter presents a study of k-NN, LI-KNN, GI-KNN for classification as classifiers. It is well known that the nearest neighbor techniques are simplest classification techniques for pattern recognition and GA can be applied for optimization. The synergy between nearest neighbor techniques and GA can be formed to utilize the attributes of both of the techniques. Using this concept we proposed a novel scheme to select a subset of features from a dataset using GA. We apply GA with a random small subset of features. The GA explores stochastically a better subset of features using various combinations of lengths and features over a number of generations. The classification accuracy of different classifiers in presence of these subsets of features is taken as the performance criteria (objective function) of GA. The proposed scheme is tested on a few UCI datasets. The performances of the k-NN, Informative k-NN (Local LI-KNN and Global GI-KNN), and LI-KNN with boosting in presence of all features and those in presence of only selected subset of features are compared with reported results. With extensive simulation study, it is observed that the proposed scheme produces a reasonably good accuracy with a reduced subset of features in these datasets [187].

6.1 Introduction

Due to automation of the day-to-day activities in our society, we depend on storage media. In most of the computer-based applications today, datasets are having a large number of patterns and relatively a much smaller number of classes. Each pattern is represented by a set of features and each pattern belongs to one of the available classes. Data mining [88] is defined as the extraction of interesting knowledge from a large database. Classification is an important step in the process of data mining. Feature analysis is an important component of classification. Two major components of feature analysis are feature selection and feature
extraction. Feature selection [34], [69] refers to selecting a subset of features from the entire set of features in a dataset. Feature extraction on the other hand, may combine or re-compute features among themselves to create a new feature. In a dataset, there can be redundant or derogatory features. These unwanted features cause confusion and increase complexity and cost of classifier. Sometimes a classifier with just sufficient number of features can produce more accurate results than that with extra features. If the feature selection makes use of information (such as class of a pattern) given before the process is applied, then the approach is called supervised. If no information is supplied a priory to grouping the patterns, the approach is called an unsupervised. A number of supervised feature selection methods exist which use Neural Networks, Fuzzy logic, k-nearest neighbor search (k-NN) algorithms.

The decision of selecting significant features is heuristic. To select most effective features from a large number of features, evolutionary computing techniques can be applied. Genetic algorithm (GA) [64], [175] is a powerful evolutionary computing technique based on the principles of evolution. GA can be applied to select features in this manner: Given an objective function $z$ (classification accuracy of a classifier), GA can be used to find a set of variables (features) for which $z$ is maximized or minimized. In this paper we attempt to investigate a subset of features using GA which produces better classification accuracy. The classifiers used in the paper are k-NN, Informative k-NN (LI-KNN and GI-KNN), and LI-KNN with boosting.

6.2 Review of Earlier Work

Data mining is defined as a multidisciplinary joint effort from databases, machine learning, and statistics, which is championing in turning mountains of data into nuggets [141]. Feature selection (FS) is an
important component of data mining as it helps in reducing dimensions of
data set but still preserving the structure of the data set. Ho T.K. [75]
combined and constructed multiple classifiers using randomly selected
features, which can achieve better performance in classification than using
the complete set of features. As an option, the selection of an optimal
feature vector can be made with an exhaustive search of all possible
subsets of features [234]. Feature selection methods can have two
categories: the wrapper method [98]; where the classification accuracy is
employed to evaluate feature subsets and the second is filter approach in
which, various measurements may be used as feature selection criteria.
The wrapper methods may perform better, but huge computational effort
is required [38]. In case of extremely large feature sets, such as the gene
(feature) set of a cDNA or microarray data, it can be enormously
expensive to apply wrapper method. A hybrid method is suggested by Liu
[123], which attempts to take advantage of both the methods by exploiting
their different evaluation criteria in different search stages. An
unsupervised algorithm [141] uses feature dependency/similarity for
redundancy reduction. The method involves partitioning of the original
feature set into some distinct subsets or clusters so that the features within
a cluster are highly similar while those in different clusters are dissimilar.
A single feature from each such cluster is then selected to constitute the
resulting reduced subset.

Use of soft computing methods like GA, fuzzy logic and Neural
Networks for FS and Feature ranking is suggested in [151], [154]. Muni et
al. [143] categorized FS method into five groups based on the evaluation
function, distance, information, dependence, consistency (all filter types),
and classifier error rate (wrapper type). Further, Setiono and Liu [191]
state that the process of FS works opposite to ID3 [166]. Instead of
selecting one attribute at a time, it starts with taking whole set of attributes and removes irrelevant attribute one by one using a three layer feed forward neural network. Basak and Pal [18] used neuro-fuzzy approach for unsupervised FS and compared with other supervised approaches. Further, Best Incremental Ranked Subset (BIRS) based algorithm [174] presents a fast search through the attribute space and any classifier can be embedded into it as evaluator. BIRS chooses a small subset of genes from the original set (0.0018% on average) with similar predictive performance to others. For very high dimensional datasets, wrapper-based methods might be computationally unfeasible, so BIRS turns out a fast technique that provides good performance in prediction accuracy. Yan et al. [230] proposed a general formulation known as graph embedding to unify a large family of algorithms - supervised or unsupervised; stemming from statistics or geometry. They proposed a new supervised dimensionality reduction algorithm called marginal Fisher analysis MFA. Recently a new unsupervised forward orthogonal search (FOS) algorithm is introduced for feature selection and ranking [221] which uses a squared correlation function as the criterion to measure the dependency between the features.

Feature selection and dimensionality reduction using GA has been widely discussed by researchers. Reymer et al. [169] used GA for dimensionality reduction. The pioneering work by Siedlecki et al [198] demonstrated evidence for the superiority of the GA compared to representative classical algorithms. Few more papers can be found in this context [25], [44], [161], [232]. Yang Song et al. [202], [203] have presented a new approach for missing data problem arising in pattern classification scenarios. In this article, they proposed a novel k-NN imputation procedure using a feature-weighted distance metric based on mutual information (MI). The MI-based distance metric is also used to
implement an effective k-NN classifier. Experimental results on both artificial and real classification datasets are provided to illustrate the efficiency and the robustness of the reported algorithm.

Present paper is inspired from the work of Yang Song et al. [202]. They have presented an improved KNN technique. The concept is as follows: in a simple k-NN, when the distribution of data is not uniform, k-NN classifiers especially with small values do not classify testing patterns accurately. With high value of k-NN (as k=45 even in few cases), the performance improves. They presented informative k-NN, according to which, it is better to consider only those points which purely belong to a class and quite far away from other class so that the boundaries in any test conditions do not misclassify a test pattern. They took all features of the datasets. In this paper authors contribute to propose a scheme by which a subset of features is capable to represent whole dataset as far as the classification accuracy of the classifier with respect to that dataset is concerned. The classification accuracies obtained due to classifiers used by Yang Song et al. [202] using entire set of features has been almost achieved with a reduced subset of features using proposed scheme in this paper. To the best perception of authors, such scheme has never been published in the literature.

### 6.3 Proposed Scheme

The GA will be used to select subset of features. The pictorial presentation of proposed scheme is shown in Figure 5.7 and 5.8. Here initial population is a representation of different subsets of features with different lengths and combinations. The fitness of the GA is taken as the classification accuracy of one of the classifiers at a time. The complete description of the proposed scheme is given in Chapter-5.
6.4 Various Classifiers and their Accuracies Used in the Proposed Scheme

In this section, the outline is given for various classifiers tested using the proposed scheme.

6.4.1 k-NN Classifier

The k-nearest neighbor (k-NN) classifier has been both a workhorse and benchmark Classifier [9], [10], [41], [162], [235]. Given a query vector $X_0$ and a set of $N$ labeled instances $\{x_i, y_i\}$ $N_1$, the task of the classifier is to predict the class label of $x_0$ on the predefined $P$ classes. The k-nearest neighbor (k-NN) classification algorithm tries to find the $k$ nearest neighbors of $X_0$ and uses a majority vote to determine the class label of $X_0$. A metric distance between any two entities is called as a notion of proximity. If two entities are in the close proximity, then they are said to belong to the same class or group. The nearest neighbor search is a method to identify entities in the same proximity in a supervised manner and is defined as “Given a collection of data points and a query point in a $d$–dimensional metric space, find the data point that is closest to the query point” [22].

For computing classification accuracy (CA) of a data set with the given number of features, we used k-NN classifier, as follows.

1. Given a data set consisting of say $P$ patterns. Generate a random sequence of patterns to shuffle classes separately if placed together in the beginning. This is to ensure that patterns in same class are not put together.

2. Create five folds of patterns in this sequence such that each fold holds $1/5$ of the patterns equally (last fold can have remaining patterns if $P$ is not divisible by 5). First four folds $f_1$, $f_2$, $f_3$, $f_4$ will
be used for training and the last fold f5 will be used for testing. Set the CA counter c_a_c to 0.

3. Compute the Euclidean distance of each pattern in f1, f2, f3, f4 from first pattern in f5, considering only the reduced features of the patterns (excluding the class). In other words, in each pattern, only featured columns will be taken into account, rest columns will be set to zero while computing distance.

4. Mark the k nearest patterns in training patterns on the basis of their Euclidean Distance from first test pattern. Note the classes of this k pattern in training set. Take the class decided by majority among these k patterns. Compare this class with the class of first testing pattern. If both do not match, add the CA counter c_a_c by 1. Repeat the exercise for all patterns in the test fold f5.

5. Compute the average classification accuracy percentage by equation

6. \[
   \text{avg } c_a_c = \frac{(100 \times (100 - c_a_c))}{(P/5)} \quad \text{\ldots (3)}
   \]

7. Now create another sequence of patterns P starting with f2, f3, f4, f5 as training and f1 as test pattern.

8. Repeat steps (iii) through (v).

9. Compute average value of c_a_c for 5 sets of folds.

6.4.2 Locally Informative k-NN (LI-KNN) Classifier

Without prior knowledge, most KNN classifiers apply Euclidean distance as the measurement of the closeness between examples. Since it has already been shown that treating the neighbors that are of low relevance as the same importance as those of high relevance could possibly degrade the performance of k-NN procedures [60], we
believe it to further explore the information exhibited by neighbors. In this section, we first propose a new distance metric that assesses the informativeness of point given a specific query point. We then proceed to use it to augment k-NN classification and advocate our first method, LI-KNN.

The Rational of informativeness is that two points are likely to share the same class label when their distance is sufficiently small, assuming the point have a uniform distribution. This idea is the same as k-NN classification. On the other hand, compared to traditional k-NN classifier that measure pair wise distances between the query point and neighbors, our metric also calculates the closeness between neighbor point, i.e., the informative point should also have a large distance form dissimilar point. This further guarantee that the locations of other informative point have the same class label maximum likelihood.

Algorithm of Classification using LI-KNN

Algorithm 1 gives the pseudo-code of LI-KNN classification. Instead of finding the informative points for each \( x_i \) by going over the entire data set, LI-KNN retrieves 1 locally informative points by first getting the K nearest neighbors (we consider the Euclidean distance here). The algorithm is presented below, details are available in [28, 29].

1. Input : ( S,K,I)
2. Target matrix : \( S = \{ x_i, y_i \}_1^N \)
3. Number of neighbors : \( K \in \{1, \ldots , N-1\} \)
4. Number of informative points : \( I \in \{1, \ldots , K\} \)
5. Initialize \( err \leftarrow 0 \)
6. For each query point \( x_i \) \((i = 1 \text{ to } N)\) do
7. Find K nearest neighbors \( X_k \) using Euclidean distance

8. Find I most informative points among K neighbors (equation (1))

9. Majority vote between the I points to determine the class label of \( x_i \)

10. If \( x_i \) is misclassified

11. \( \text{err} \leftarrow \text{err} + \frac{1}{N} \)

12. End if

13. End for

14. Output \( \text{err} \)

6.4.3 Globally Informative –k-NN Classifier

The LI-KNN algorithm classifier each individual query point by learning informative points separately, however, the informativeness of those neighbors are then discarded without being utilized for other query point. Indeed, in most scenarios, different queries Q may yield different informative points. However it is reasonable to expect that some points are more informative than others, i.e. they could be informative neighbor for several different points. As a result, it would seem reasonable to put more emphasis on those points that are globally informative. Since it has been shown that k-NN classification can be improved by learning from training examples a distance metric, in this section we enhance the power of the informativeness metric and propose a boosting –like iterative method, namely a globally informative k-NN (GI-KNN) that aims to learn the best weighting for points within the entire training set.

Algorithm of Pattern Classification using GI-KNN

1. Input : ( T,K,I,M)

2. Training set : \( T=\{x,y\} \in \mathbb{R}^{N \times P} \)
3. Number of neighbors : \( k \in \{1, \ldots, N-1\} \)

4. Number of informative points : \( I > e \in \{1, \ldots, K\} \)

5. Number of Iteration : \( M \in \mathbb{R} \)

6. Initialization : \( A = \{1, \ldots, l\} \in \mathbb{R}^{N \times l} \) [the weight vector]

7. For \( m = 1 \) to \( M \) do

8. For each query point \( x_i (i=1 \) to \( N) \) do

9. \[ D_{A(x_i, x)} = \frac{D_{A(x_i, x)}}{A} \] [calculate the weight distance]

10. \( N_m(x_i) \) \( \leftarrow I \) most informative points according to \( D_A(x_i, x) \)

11. \[ e'_m = A(x_i).E_A[N_m(x_i)] = A(x_i).\frac{1}{I} \sum_{i=1}^{l} A(N_m(i)) \]

12. \[ C'_m = \frac{1}{2} (1 + \tanh(\tau(\epsilon'_m - \frac{1}{2}))) \]

13. \[ e(x_i, x_j) = \begin{cases} C'_m & \text{if } y_i \neq y_j \\ 0 & \text{if } y_i = y_j \end{cases} \]

14. if point \( x_i \) is classified incorrectly [update the neighbors’ weight]

15. \( \text{err}_m \leftarrow \text{err}_m + \frac{1}{N} \)

16. for each \( x_i (i \in N_m(x_i)) \) do

17. \( A(x_i) \leftarrow A(x_i).\exp(-e(x_i, x_j)) \)

18. end for

19. renormalizes \( A \) so that \( \sum_{i=1}^{N} A(i) = N \)

20. End for

21. \( \xi_m \leftarrow \text{err}_m - \text{err}_{m-1} \)

22. End for

23. Output: the weight vector \( A \)
6.4.4 Boosting with LI-KNN Classifier

Boosting assigns high weights to points that are misclassified in the current step, so that the weak learner can attempt to fix the errors in future iteration. In GI-KNN classification, the objective is to finds globally informative points thus higher weights are given to the neighbors that seldom makes wrong predictions. Notice that the weight of the query point remains unchanged at that time, because the weight is updated for a specific point if and only if it is chosen to be one of the informative points for Q.

Another difference from boosting is that the objective of the Boosting training process is to find a committee of discriminant classifier that combine the Weak learners, while GI-KNN tries to learn a query-based distance metric by focusing on finding the best weight vector for each training instance so that the misclassification rate of the training examples could be minimized.

6.5 Simulation Studies and Results Analysis

The proposed scheme was simulated on an Intel Core to Duo PIII machine. The results are shown in Table-6.1. The table is broadly divided in four columns. Different datasets are listed in first column. The second column shows the number of patterns (N) in individual dataset, the number of features (D) and number of classes (C) in the respective dataset. The third column contains the performance of the proposed scheme while the fourth column expresses the results already published as a comparison. The scheme was applied on six datasets. The results of five datasets are compared with those available in the literature [202]. The classification accuracy of a dataset obtained using k-NN, LI-KNN, GI-KNN and LI-KNN with boosting based classifiers in presence of all
features in that dataset is shown in column 4. The third column contains the classification accuracy of a dataset in presence of different reduced feature sets as obtained using GA for the same classifiers. From this table it is noted that the performance of scheme is better in case of iris, wine, glass and wbc datasets. We note that these datasets with 3 (4), 10 (13), 6 (9), 5 (9) features (inside brackets total number of features) are capable to produce the same CA (or even better) for given classifiers under the same computational platform than those with all features. In case of ionosphere, merely 25 (out of 34) features produce competitive CA for given classifiers compared to those reported in the literature. The interesting point to note here is that extra features produced smaller CA. The thyroid data set is simulated for computing CA and it is found that with only 3 features (out of 5) we are able to get a CA of over 90% in almost all cases except simple k-NN. Table-6.2 contains details of GA applied in the scheme along with time taken by each classification algorithm. The pictorial representation of comparison between best of the proposed technique and the conventional nearest neighbor (k-NN) is shown in Figure-6.1, Figure-6.2, Figure-6.3, Figure-6.4, Figure-6.5 and Figure-6.6. The summary of the datasets are shown in Table-6.3 and the details of the datasets used are given in Appendix-A.
<table>
<thead>
<tr>
<th>Classification Accuracy in the Published Literature</th>
<th>%</th>
<th>%</th>
<th>%</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boosting</td>
<td>90.60 (135)</td>
<td>82.20 (135)</td>
<td>84.00 (135)</td>
<td>90.60 (135)</td>
</tr>
<tr>
<td>1-L-KNN</td>
<td>90.40 (135)</td>
<td>88.20 (135)</td>
<td>89.00 (135)</td>
<td>90.60 (135)</td>
</tr>
<tr>
<td>1-KNN</td>
<td>90.40 (135)</td>
<td>88.20 (135)</td>
<td>89.00 (135)</td>
<td>90.60 (135)</td>
</tr>
<tr>
<td>L-KNN</td>
<td>90.40 (135)</td>
<td>88.20 (135)</td>
<td>89.00 (135)</td>
<td>90.60 (135)</td>
</tr>
<tr>
<td>Total Features</td>
<td>90.40 (135)</td>
<td>88.20 (135)</td>
<td>89.00 (135)</td>
<td>90.60 (135)</td>
</tr>
</tbody>
</table>

| % | % | % | % |
|---------------------------------------------------|---|---|---|---|
| Boosting                                          | 89.00 (135) | 85.00 (135) | 86.80 (135) | 87.60 (135) |
| 1-L-KNN                                          | 90.40 (135) | 88.20 (135) | 89.00 (135) | 90.60 (135) |
| 1-KNN                                            | 90.40 (135) | 88.20 (135) | 89.00 (135) | 90.60 (135) |
| L-KNN                                            | 90.40 (135) | 88.20 (135) | 89.00 (135) | 90.60 (135) |
| Total Features                                   | 90.40 (135) | 88.20 (135) | 89.00 (135) | 90.60 (135) |

| % | % | % | % |
|---------------------------------------------------|---|---|---|---|
| Boosting                                          | 87.29 (9) | 90.34 (9) | 91.38 (9) | 92.72 (9) |
| 1-L-KNN                                          | 90.40 (9) | 93.50 (9) | 95.00 (9) | 96.50 (9) |
| 1-KNN                                            | 90.40 (9) | 93.50 (9) | 95.00 (9) | 96.50 (9) |
| L-KNN                                            | 90.40 (9) | 93.50 (9) | 95.00 (9) | 96.50 (9) |
| Total Features                                   | 90.40 (9) | 93.50 (9) | 95.00 (9) | 96.50 (9) |

<table>
<thead>
<tr>
<th>C</th>
<th>D</th>
<th>N</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>150</td>
<td>214</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>178</td>
<td>683</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>214</td>
<td>351</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>4</td>
<td>315</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Data Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wine Data</td>
<td>Wine Data</td>
</tr>
<tr>
<td>Data Set</td>
<td>Data Set</td>
</tr>
<tr>
<td>DBN</td>
<td>DBN</td>
</tr>
<tr>
<td>Wine Class Data</td>
<td>Wine Class Data</td>
</tr>
<tr>
<td>Toy</td>
<td>Toy</td>
</tr>
<tr>
<td>Dataset</td>
<td>Dataset</td>
</tr>
</tbody>
</table>
### Table 6.2: Details of the datasets and GA applied for feature selection

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Classification Method</th>
<th>Population size (crossovers × features)</th>
<th>Crossover Probability</th>
<th>Mutation</th>
<th>Generation</th>
<th>Time Taken second</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris Data</td>
<td>k-NN, IKNN, Boosting</td>
<td>20/4</td>
<td>.5</td>
<td>.3</td>
<td>50</td>
<td>35 sec</td>
</tr>
<tr>
<td>Wine</td>
<td>k-NN, IKNN, Boosting</td>
<td>10000/13</td>
<td>.5</td>
<td>.3</td>
<td>50</td>
<td>1.25 sec</td>
</tr>
<tr>
<td>Glass</td>
<td>k-NN, IKNN, Boosting</td>
<td>1500/10</td>
<td>.5</td>
<td>.3</td>
<td>50</td>
<td>54 sec</td>
</tr>
<tr>
<td>WBC</td>
<td>k-NN, IKNN, Boosting</td>
<td>700/9</td>
<td>.5</td>
<td>.3</td>
<td>50</td>
<td>48 sec</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>k-NN, IKNN, Boosting</td>
<td>30000/34</td>
<td>.5</td>
<td>.3</td>
<td>50</td>
<td>3.75 sec</td>
</tr>
<tr>
<td>Thyroid</td>
<td>k-NN, IKNN, Boosting</td>
<td>40/5</td>
<td>.5</td>
<td>.3</td>
<td>50</td>
<td>35 sec</td>
</tr>
</tbody>
</table>

### Table 6.3: Summary of Data Sets Used

<table>
<thead>
<tr>
<th>Name of data set</th>
<th>#classes</th>
<th># Features</th>
<th>Size of Data set (With Class)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris</td>
<td>3</td>
<td>4</td>
<td>150 (50+50+50)</td>
</tr>
<tr>
<td>Thyroid</td>
<td>3</td>
<td>5</td>
<td>215 (150+35+30)</td>
</tr>
<tr>
<td>WBC</td>
<td>2</td>
<td>9</td>
<td>683 (444+239)</td>
</tr>
<tr>
<td>Glass</td>
<td>6</td>
<td>9</td>
<td>214 (70+76+17+13+2:29)</td>
</tr>
<tr>
<td>Wine</td>
<td>3</td>
<td>13</td>
<td>178 (59+72+47)</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>2</td>
<td>34</td>
<td>351 (225+126)</td>
</tr>
</tbody>
</table>
**Figure 6.1**: Pictorial presentation of comparative study on IRIS data set

**Figure 6.2**: Pictorial presentation of comparative study on Wine data set
Figure 5.19: HMBC Spectrum of Compound 6
**Glass Data Set**

![Graph showing classification accuracy comparison between Proposed Technique and Conventional Technique for Glass data set.]

**Figure 6.3**: Pictorial presentation of comparative study on Glass data set.

**WBC Data Set**

![Graph showing classification accuracy comparison between Proposed Technique and Conventional Technique for WBC data set.]

**Figure 6.4**: Pictorial presentation of comparative study on WBC data set.
Figure 5.20: FAB'-Mass Spectrum of Compound 7
Figure 6.5: Pictorial presentation of comparative study on Ionosphere data set

Figure 6.6: Pictorial presentation of comparative study on Thyroid data set